10/621,450

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年　７月１４日

出願番号
Application Number: 特願２００３－１９６２２８

[ST. 10/C]: [ＪＰ２００３－１９６２２８]

出願人
Applicant(s): 株式会社リコー

２００３年　７月２８日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号　出証特２００３－３０５９７８６

【書類名】　特許願
【整理番号】　0305119
【提出日】　平成15年 7月14日
【あて先】　特許庁長官　今井　康夫　殿
【国際特許分類】　G03G 21/00　370
【発明の名称】　画像形成装置及びラッピング処理方法並びにプログラム
【請求項の数】　34
【発明者】
　【住所又は居所】　東京都大田区中馬込１丁目３番６号　株式会社リコー内
　【氏名】　大石　勉
【発明者】
　【住所又は居所】　東京都大田区中馬込１丁目３番６号　株式会社リコー内
　【氏名】　中川　克彦
【発明者】
　【住所又は居所】　東京都大田区中馬込１丁目３番６号　株式会社リコー内
　【氏名】　秋吉　邦洋
【発明者】
　【住所又は居所】　東京都大田区中馬込１丁目３番６号　株式会社リコー内
　【氏名】　田中　浩行
【特許出願人】
　【識別番号】　000006747
　【氏名又は名称】　株式会社リコー
【代理人】
　【識別番号】　100070150
　【弁理士】
　【氏名又は名称】　伊東　忠彦
【先の出願に基づく優先権主張】
　【出願番号】　特願2002-211685
　【出願日】　平成14年 7月19日

【手数料の表示】

【予納台帳番号】　002989

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9911477

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　画像形成装置及びラッピング処理方法並びにプログラム

【特許請求の範囲】

【請求項１】　画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置において、

アプリケーションから呼び出される関数を変換し、変換後の関数を用いてコントロールサービスへの関数呼び出しを行うラッピング処理手段を備えたことを特徴とする画像形成装置。

【請求項２】　前記ラッピング処理手段は、前記アプリケーションが前記コントロールサービスに対して使用する関数と、この関数に対応してコントロールサービスが有する関数との間でバージョン差がある場合に前記関数の変換を行う請求項１に記載の画像形成装置。

【請求項３】　前記ラッピング処理手段は、前記コントロールサービスが有する関数のバージョンが変更されたことを示す情報を参照することにより、前記バージョン差があることを判断する請求項２に記載の画像形成装置。

【請求項４】　前記ラッピング処理手段は、前記アプリケーションが前記コントロールサービスに対して使用する関数と、当該関数に対応するコントロールサービス側の関数との間で関数の数又は引数の数が異なる場合に、ダミーの関数又はダミーの引数を補填する変換を行う請求項１に記載の画像形成装置。

【請求項５】　画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置において、

前記コントロールサービスが送信するメッセージの中から前記アプリケーションに通知するメッセージを取捨選択するラッピング処理手段を備えたことを特徴とする画像形成装置。

【請求項６】　前記ラッピング処理手段は、前記アプリケーションに対して通知しない特定のメッセージを記録した情報を参照することにより、前記アプリケーションに対するメッセージを取捨選択する請求項５に記載の画像形成装置。

【請求項７】　前記コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスを更に備え、

前記ラッピング処理手段は、前記仮想アプリケーションサービスに含まれる請求項１ないし６のうちいずれか１項に記載の画像形成装置。

【請求項８】　前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョンが、前記仮想アプリケーションサービスがサポートできる所定の範囲内であるか否かを判断するバージョン管理手段を更に有する請求項７に記載の画像形成装置。

【請求項９】　前記バージョン管理手段は、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記バージョンが前記所定の範囲内であるか否かを判断する請求項８に記載の画像形成装置。

【請求項１０】　前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数のバージョンが、前記仮想アプリケーションサービスがサポートできる所定の範囲内であるか否かを、関数単位に判断するバージョン管理手段を更に有する請求項７に記載の画像形成装置。

【請求項１１】　前記バージョン管理手段は、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数毎のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記関数のバージョンが前記所定の範囲内であるか否かを関数単位に判断する請求項１０に記載の画像形成装置。

【請求項１２】　画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置が実行する処理の方法であって、

アプリケーションから呼び出される関数を変換し、変換後の関数を用いてコントロールサービスへの関数呼び出しを行うラッピング処理ステップを有することを特徴とする方法。

【請求項１３】　前記ラッピング処理ステップにおいて、前記アプリケーションが前記コントロールサービスに対して使用する関数と、この関数に対応してコントロールサービスが有する関数との間でバージョン差がある場合に関数の変換を行う請求項１２に記載の方法。

【請求項１４】　前記ラッピング処理ステップにおいて、前記コントロールサービスが有する関数のバージョンが変更されたことを示す情報を参照することにより、前記バージョン差があることを判断する請求項１３に記載の方法。

【請求項１５】　前記ラッピング処理ステップにおいて、前記アプリケーションが前記コントロールサービスに対して使用する関数と、当該関数に対応するコントロールサービス側の関数との間で関数の数又は引数の数が異なる場合に、ダミーの関数又はダミーの引数を補填する変換を行う請求項１２に記載の方法。

【請求項１６】　画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置が実行する処理の方法であって、

前記コントロールサービスが送信するメッセージの中から前記アプリケーションに通知するメッセージを取捨選択するラッピング処理ステップを有することを特徴とする方法。

【請求項１７】　前記ラッピング処理ステップにおいて、前記アプリケーションに対して通知しない特定のメッセージを記録した情報を参照することにより、前記アプリケーションに対するメッセージを取捨選択する請求項１６に記載の方法。

【請求項１８】　前記画像形成装置は、前記コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスを更に備え、前記ラッピング処理ステップは、前記仮想アプリケーションサービスが実行する請求項１２ないし１７のうちいずれか１項に記載の方法。

【請求項１９】　前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョンが、前記仮想アプリケーションサービスがサポートできる所定の範囲内であるか否かを判断するバージョン管理ステッ

プを更に有する請求項１８に記載の方法。

【請求項２０】　前記バージョン管理ステップにおいて、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記バージョンが前記所定の範囲内であるか否かを判断する請求項１９に記載の方法。

【請求項２１】　前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数のバージョンが、前記仮想アプリケーションサービスがサポートできる所定の範囲内であるか否かを、関数単位に判断するバージョン管理ステップを更に有する請求項１８に記載の方法。

【請求項２２】　前記バージョン管理ステップにおいて、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数毎のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記関数のバージョンが前記所定の範囲内であるか否かを関数単位に判断する請求項２１に記載の方法。

【請求項２３】　画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置を、

アプリケーションから呼び出される関数を変換し、変換後の関数を用いてコントロールサービスへの関数呼び出しを行うラッピング処理手段として機能させるプログラム。

【請求項２４】　前記ラッピング処理手段は、前記アプリケーションが前記コントロールサービスに対して使用する関数と、この関数に対応してコントロールサービスが有する関数との間でバージョン差がある場合に関数の変換を行う請求項２３に記載のプログラム。

【請求項２５】　前記ラッピング処理手段は、前記コントロールサービスが有する関数のバージョンが変更されたことを示す情報を参照することにより、前記バージョン差があることを判断する請求項２４に記載のプログラム。

【請求項２６】　前記ラッピング処理手段は、前記アプリケーションが前記

コントロールサービスに対して使用する関数と、当該関数に対応するコントロールサービス側の関数との間で関数の数又は引数の数が異なる場合に、ダミーの関数又はダミーの引数を補填する変換を行う請求項２３に記載のプログラム。

【請求項２７】　画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置を、

前記コントロールサービスが送信するメッセージの中から前記アプリケーションに通知するメッセージを取捨選択するラッピング処理手段として機能させるプログラム。

【請求項２８】　前記ラッピング処理手段は、前記アプリケーションに対して通知しない特定のメッセージを記録した情報を参照することにより、前記アプリケーションに対するメッセージを取捨選択する請求項２７に記載のプログラム。

【請求項２９】　前記プログラムは、前記コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスのプログラムに含まれる請求項２３ないし２８のうちいずれか１項に記載のプログラム。

【請求項３０】　前記画像形成装置を、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョンが、前記仮想アプリケーションサービスがサポートできる所定の範囲内であるか否かを判断するバージョン管理手段として更に機能させる請求項２９に記載のプログラム。

【請求項３１】　前記バージョン管理手段は、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記バージョンが前記所定の範囲内であるか否かを判断する請求項３０に記載のプログラム。

【請求項３２】　前記画像形成装置を、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数のバージョンが、前記仮想アプリ

ケーションサービスがサポートできる所定の範囲内であるか否かを、関数単位に判断するバージョン管理手段として更に機能させる請求項２９に記載のプログラム。

【請求項３３】　前記バージョン管理手段は、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数毎のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記関数のバージョンが前記所定の範囲内であるか否かを関数単位に判断する請求項３２に記載のプログラム。

【請求項３４】　請求項２３ないし３３のうちいずれか１項に記載のプログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

この発明は、コピー、プリンタ、スキャナおよびファクシミリなどの画像形成処理にかかるユーザサービスを提供する画像形成装置に関し、特に、アプリケーションとアプリケーションが利用するサービスとの間でバージョン差が生じた場合に、バージョン差を吸収できる画像形成装置に関するものである。

【０００２】

【従来の技術】

近年では、プリンタ、コピー、ファクシミリ、スキャナなどの各装置の機能を１つの筐体内に収納した画像形成装置（以下、「複合機」という。）が一般的に知られている。この複合機は、１つの筐体内に表示部、印刷部および撮像部などを設けるとともに、プリンタ、コピーおよびファクシミリ装置にそれぞれ対応する３種類のソフトウェアを設け、これらのソフトウェアを切り替えることによって、当該装置をプリンタ、コピー、スキャナ又はファクシミリ装置として動作させるものである。

【０００３】

このような従来の複合機では、プリンタ、コピー、ファクシミリ、スキャナなどの各機能単位でアプリケーションプログラムが起動され、ハードウェア資源に

アクセスする機能を持ち合わせている。その際、アプリケーションプログラムが前提としているオペレーティングシステム（ＯＳ）のバージョンと、実際のオペレーティングシステム（ＯＳ）のバージョンとが同じことが前提となるが、例えば、ＯＳをバージョンアップしてアプリケーションとの間でバージョン差が生じた場合、今まで使えていた機能が使えなくなったり、アプリケーションそのものが起動しなくなったりすることがある。

【０００４】

このため、従来の複合機では、ＯＳなどをバージョンアップすると、それに伴ってアプリケーションの方もコンパイルし直すことで、常に対応したバージョン関係にあることが要請されている。

【０００５】

【特許文献１】

特開２００２－８２８０６号公報

【０００６】

【発明が解決しようとする課題】

ところで、このような従来の複合機では、プリンタ、コピー、スキャナおよびファクシミリ装置に対応するソフトウェアがそれぞれ別個に設けられているため、各ソフトウェアの開発に多大の時間を要する。このため、出願人は、表示部、印刷部および撮像部などの画像形成処理で使用されるハードウェア資源を有し、プリンタ、コピー又はファクシミリなどの各ユーザサービスにそれぞれ固有の処理を行うアプリケーションを複数搭載し、これらのアプリケーションとハードウェア資源との間に介在して、ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とするハードウェア資源の管理、実行制御並びに画像形成処理を行う各種コントロールサービスからなるプラットホームを備えた画像形成装置（複合機）を発明した。

【０００７】

このような新規な複合機では、アプリケーションとコントロールサービスとを別個に設けているため、複合機の出荷後にユーザもしくは第三者であるサードベンダが新規なアプリケーションを開発して複合機に搭載することが可能となり、

これによって多種多様な機能を提供することが可能となる。

【０００８】

このような新規な複合機では、アプリケーションの少なくとも２つが共通に必要とするサービスを提供するコントロールサービスをアプリケーションと別個に設けた構成となっているため、新規アプリケーションを開発する場合には、各種コントロールサービスとのプロセス間通信を実現する処理をソースコードで記述する必要がある。しかしながら、新規アプリケーションを開発する場合、各コントロールサービスが提供する関数やメッセージなどを正確に把握した上で、予め規定された手順で記述しなければならず、各コントロールサービスやアプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェース（ＡＰＩ）をデバッグや機能追加によって繰り返しバージョンアップすると、ベンダーはどのバージョンに合わせてアプリケーション開発すればよいかが非常に分かり難くなり、アプリケーションの開発が難しくなる。また、バージョンｎ差のあるアプリケーションを実行させることにより、システムに影響を及ぼす恐れもある。このことは、固定的な複数の機能を寄せ集めた従来の複合機では問題にならなかった新規な課題である。

【０００９】

また、各コントロールサービスとアプリケーションのインタフェース（ＡＰＩ）のすべてを、新規アプリケーションを開発するサードベンダなどの第三者に開示することは、プログラムの秘匿性、システムの安全性の観点から好ましくない。

【００１０】

この発明は上記に鑑みてなされたもので、バージョンアップ等によりアプリケーションが使用している関数とコントロールサービス側の関数とで差異が生じた場合でも、アプリケーションのソースプログラムを書き換えることなく、関数呼び出しを行うことを可能とする技術を提供することを目的とする。また、コントロールサービスからアプリに通知されるメッセージを秘匿する技術を提供することを目的とする。

【００１１】

【課題を解決するための手段】

上記目的を達成するため、請求項１にかかる発明は、画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置において、アプリケーションから呼び出される関数を変換し、変換後の関数を用いてコントロールサービスへの関数呼び出しを行うラッピング処理手段を備える。

【００１２】

本発明によれば、アプリケーションから呼び出された関数を変換して、関数呼び出しを行うので、アプリケーションが使用している関数とコントロールサービス側の関数とで差異が生じた場合でも、アプリケーションのソースプログラムを書き換えることなく、関数呼び出しを行うことが可能となる。

【００１３】

請求項２に記載の発明は、請求項１の記載において、前記ラッピング処理手段は、前記アプリケーションが前記コントロールサービスに対して使用する関数と、コントロールサービスが有する関数との間でバージョン差がある場合に関数の変換を行う請求項１に記載の画像形成装置。

【００１４】

本発明によれば、例えば、コントロールサービス側の関数のバージョンアップがあり、バージョン差が生じている場合に関数の変換を行う。これにより、バージョン不整合による不具合発生を防止できる。
することができる。

【００１５】

請求項３に記載の発明は、請求項２の記載において、前記ラッピング処理手段は、前記コントロールサービスが有する関数のバージョンが変更されたことを示す情報を参照することにより、前記バージョン差があることを判断する。これによりバージョン差があること正確に判断できる。

【００１６】

請求項４に記載の発明は、請求項１の記載において、前記ラッピング処理手段は、前記アプリケーションが前記コントロールサービスに対して使用する関数と

、当該関数に対応するコントロールサービス側の関数との間で関数の数又は引数の数が異なる場合に、ダミーの関数又はダミーの引数を補填する変換を行う。

【００１７】

アプリケーションが前記コントロールサービスに対して使用する関数と、当該関数に対応するコントロールサービス側の関数との間で関数の数又は引数の数が異なる場合には、関数呼び出しを行うことができなくなるが、本発明によれば、増加した関数や関数の引数に対応するダミーの関数又は関数の引数を補填することによって関数呼び出しが可能となる。

【００１８】

請求項５に記載の発明は、画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置において、前記コントロールサービスが送信するメッセージの中から前記アプリケーションに通知するメッセージを取捨選択するラッピング処理手段を備えたものである。

【００１９】

本発明によれば、第三者に対してはコントロールサービスとの間のインタフェースを隠蔽することができる。

【００２０】

請求項６に記載の発明は、請求項５の記載において、前記ラッピング処理手段は、前記アプリケーションに対して通知しない特定のメッセージを記録した情報を参照することにより、前記アプリケーションに対するメッセージを取捨選択する。

【００２１】

本発明によれば、アプリケーションに対して通知しない特定のメッセージを記録した情報を参照するようにしたので、当該情報に所望のメッセージを記録しておくことにより、容易にメッセージを取捨選択を行うことが可能となる。

【００２２】

請求項７に記載の発明は、請求項１ないし６のうちいずれか１項の記載において、前記コントロールサービスをサーバとしたクライアントプロセスとして動作

し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスを更に備え、前記ラッピング処理手段は、前記仮想アプリケーションサービスに含まれるというものである。本発明も、請求項１等と同様の効果を奏する。

【００２３】

請求項８に記載の発明は、請求項７の記載のおいて、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョンが、前記仮想アプリケーションサービスがサポートできる所定の範囲内であるか否かを判断するバージョン管理手段を更に有するものである。

【００２４】

本発明によれば、アプリケーションが仮想アプリケーションサービスに対して使用する関数全体のバージョンが、仮想アプリケーションサービスのサポート範囲内か否かを判断できるため、例えば、アプリケーションを実際に使用する前にこの判断を行うことにより、アプリケーションが仮想アプリケーションサービス上で使用できるもの否かを判断でき、障害の発生を未然に防止することができる。

【００２５】

請求項９に記載の発明は、請求項８の記載において、前記バージョン管理手段は、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数全体のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記バージョンが前記所定の範囲内であるか否かを判断するものである。本発明によれば、容易に前記バージョンが前記所定の範囲内であるか否かを判断できる。

【００２６】

請求項１０に記載の発明は、請求項７の記載において、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数のバージョンが、前記仮想アプリケーションサービスがサポートできる所定の範囲内であるか否かを、関数単位に判断するバージョン管理手段を更に有するものである。

【００２７】

本発明によれば、関数単位に判断できるので、例えば、アプリケーションが使用していない関数がバージョンアップされても、このことに影響を受けずに判断を行うことができる。また、サポートが不可能なバージョン差がある関数を容易に特定することができる。

【００２８】

請求項１１に記載の発明は、請求項１０の記載において、前記バージョン管理手段は、前記アプリケーションが前記仮想アプリケーションサービスに対して使用する関数毎のバージョン情報を前記アプリケーションから取得し、前記所定の範囲を記録した情報を参照することにより、前記関数のバージョンが前記所定の範囲内であるか否かを関数単位に判断する。本発明によれば、容易に上記の判断を行うことができる。

【００２９】

請求項１２～２２に記載の発明は、上記の画像形成装置における処理の方法の発明である。請求項２３～３３に記載の発明は、上記の画像形成装置における処理の手順を画像形成装置に実行させるためのプログラムである。また、請求項３４に記載の発明は、そのプログラムを格納したコンピュータ読み取り可能な記録媒体である。これらの発明によっても上記の画像形成装置の発明と同様の効果を奏する。

【００３０】

【発明の実施の形態】

以下に添付図面を参照して、本発明の好適な実施の形態を詳細に説明する。

【００３１】

（実施の形態１）

図１は、この発明の実施の形態１である画像形成装置（以下、「複合機」という）の構成を示すブロック図である。図１に示すように、複合機１００は、白黒レーザプリンタ（B&W LP）１０１と、カラーレーザプリンタ（Color LP）１０２と、スキャナ、ファクシミリ、ハードディスク、メモリ、ネットワークインタフェースなどのハードウェアリソース１０３を有するとともに、プラットホーム１２０とアプリケーション１３０と仮想アプリケーションサービス（ＶＡＳ：Ｖｉ

ｒｔｕａｌ　Ａｐｐｌｉｃａｔｉｏｎ　Ｓｅｒｖｉｃｅ）１４０から構成されるソフトウェア群１１０を備えている。

【００３２】

仮想アプリケーションサービス（ＶＡＳ）１４０は、アプリケーション１３０とプラットホーム１２０の間に配置される。ＶＡＳ１４０は、アプリから見るとプラットホーム１２０のサービス層として認識され、サービス層から見るとアプリとして認識されるように登録される。また、ＶＡＳ１４０は、アプリに対してはサーバプロセスとして動作し、各コントロールサービスに対してはクライアントプロセスとして動作する。

【００３３】

このＶＡＳ１４０の第１の基本機能は、アプリケーションがコントロールサービスの機能を利用するために使用しているＡＰＩとしての関数と、コントロールサービスが提供するＡＰＩとしての関数との間でのバージョン差を吸収する機能である。この機能により、バージョン差があったとしても、アプリケーション１３０を再コンパイルすることなくそのまま関数呼び出しを行うことができる。また、コントロールサービスからのメッセージを取捨選択することによってプラットホーム１２０を隠蔽する機能もある。これらをラッピング機能という。

【００３４】

ＶＡＳ１４０の第２の基本機能は、アプリがＶＡＳ１４０に対して使用する関数のバージョンと、ＶＡＳ１４０が有する関数のバージョンとのバージョン差を検出し、その検出されたバージョン差がＶＡＳ１４０によってサポート可能な範囲か否かを判定して、その判定結果をアプリに通知する機能を有している。これにより、例えばアプリ動作前にバージョンが適切かどうかを判断できる。これをバージョン管理機能という。

【００３５】

なお、ＶＡＳ１４０のプログラムを、例えば、ＳＤ（Secure Digital）カードに格納して、そこからＶＡＳ１４０を起動することも可能である。また、ネットワーク接続されたサーバにＶＡＳ１４０のプログラムを格納しておき、そこからネットワーク経由でＶＡＳ１４０を複合機１００にインストール、もしくは、起

動することが可能である。

【００３６】

プラットホーム１２０は、アプリケーションからの処理要求を解釈してハードウェア資源の獲得要求を発生させるコントロールサービスと、一又は複数のハードウェア資源の管理を行い、コントロールサービスからの獲得要求を調停するシステムリソースマネージャ（ＳＲＭ）１２３と、汎用ＯＳ１２１とを有する。

【００３７】

コントロールサービスは、複数のサービスモジュールから形成され、ＳＣＳ（システムコントロールサービス）１２２と、ＥＣＳ（エンジンコントロールサービス）１２４と、ＭＣＳ（メモリコントロールサービス）１２５と、ＯＣＳ（オペレーションパネルコントロールサービス）１２６と、ＦＣＳ（ファックスコントロールサービス）１２７と、ＮＣＳ（ネットワークコントロールサービス）１２８とから構成される。なお、このプラットホーム１２０の各サービスは、予め定義された関数により前記アプリケーション１３０から処理要求を受信可能とするアプリケーションプログラムインタフェース（ＡＰＩ）を有している。

【００３８】

汎用ＯＳ１２１は、ＵＮＩＸ（登録商標）などの汎用オペレーティングシステムであり、プラットホーム１２０並びにアプリケーション１３０の各ソフトウェアをそれぞれプロセスとして並列実行する。

【００３９】

ＳＲＭ１２３のプロセスは、ＳＣＳ１２２とともにシステムの制御およびリソースの管理を行うものである。ＳＲＭ１２３のプロセスは、スキャナ部やプリンタ部などのエンジン、メモリ、ＨＤＤファイル、ホストＩ／Ｏ（セントロＩ／Ｆ、ネットワークＩ／Ｆ、ＩＥＥＥ１３９４　Ｉ／Ｆ、ＲＳ２３２Ｃ　Ｉ／Ｆなど）のハードウェア資源を利用する上位層からの要求にしたがって調停を行い、実行制御する。

【００４０】

具体的には、このＳＲＭ１２３は、要求されたハードウェア資源が利用可能であるか（他の要求により利用されていないかどうか）を判断し、利用可能であれ

ば要求されたハードウェア資源が利用可能である旨を上位層に伝える。また、ＳＲＭ１２３は、上位層からの要求に対してハードウェア資源の利用スケジューリングを行い、要求内容（例えば、プリンタエンジンにより紙搬送と作像動作、メモリ確保、ファイル生成など）を直接実施している。

【００４１】

ＳＣＳ１２２のプロセスは、アプリ管理、操作部制御、システム画面表示、ＬＥＤ表示、リソース管理、割り込みアプリ制御などを行う。

【００４２】

ＥＣＳ１２４のプロセスは、白黒レーザプリンタ（B&W LP）１０１、カラーレーザプリンタ（Color LP）１０２、スキャナ、ファクシミリなどからなるハードウェアリソース１０３のエンジンの制御を行う。

【００４３】

ＭＣＳ１２５のプロセスは、画像メモリの取得および解放、ハードディスク装置（ＨＤＤ）の利用、画像データの圧縮および伸張などを行う。

【００４４】

ＦＣＳ１２７のプロセスは、システムコントローラの各アプリ層からＰＳＴＮ／ＩＳＤＮ網を利用したファクシミリ送受信、ＢＫＭ（バックアップＳＲＡＭ）で管理されている各種ファクシミリデータの登録／引用、ファクシミリ読みとり、ファクシミリ受信印刷、融合送受信を行うためのＡＰＩを提供する。

【００４５】

ＮＣＳ１２８のプロセスは、ネットワークＩ／Ｏを必要とするアプリケーションに対して共通に利用できるサービスを提供するためのプロセスであり、ネットワーク側から各プロトコルによって受信したデータを各アプリケーションに振り分けたり、アプリケーションからデータをネットワーク側に送信する際の仲介を行う。具体的には、ftpd、httpd、lpd、snmpd、telnetd、smtpdなどのサーバデーモンや、同プロトコルのクライアント機能などを有している。

【００４６】

ＯＣＳ１２６のプロセスは、オペレータ（ユーザ）と本体制御間の情報伝達手段となるオペレーションパネル（操作パネル）の制御を行う。ＯＣＳ１２６は、

オペレーションパネルからのキー押下をキーイベントとして取得し、取得したキーに対応したキーイベント関数をＳＣＳ１２２に送信するＯＣＳプロセスの部分と、アプリケーション１３０又はコントロールサービスからの要求によりオペレーションパネルに各種画面を描画出力する描画関数やその他オペレーションパネルに対する制御を行う関数などが予め登録されたＯＣＳライブラリの部分とから構成される。このＯＣＳライブラリは、アプリケーション１３０およびコントロールサービスの各モジュールにリンクされて実装されている。なお、ＯＣＳ１２６のすべてをプロセスとして動作させるように構成しても良く、あるいはＯＣＳ１２６のすべてをＯＣＳライブラリとして構成しても良い。

【００４７】

アプリケーション１３０は、ページ記述言語（ＰＤＬ）、ＰＣＬおよびポストスクリプト（ＰＳ）を有するプリンタ用のアプリケーションであるプリンタアプリ１１１と、コピー用アプリケーションであるコピーアプリ１１２と、ファクシミリ用アプリケーションであるファックスアプリ１１３と、スキャナ用アプリケーションであるスキャナアプリ１１４と、ネットワークファイル用アプリケーションであるネットファイルアプリ１１５と、工程検査用アプリケーションである工程検査アプリ１１６とを有している。これらの各アプリは、その起動時にＶＡＳ１４０に対して自プロセスのプロセスＩＤとともにアプリ登録要求メッセージを送信し、アプリ登録要求メッセージを受信したＶＡＳ１４０によって、起動したアプリに対する登録が行われるようになっている。

【００４８】

アプリケーション１３０の各プロセス、コントロールサービスの各プロセスは、関数呼び出しとその戻り値送信およびメッセージの送受信によってプロセス間通信を行いながら、コピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるユーザサービスを実現している。

【００４９】

このように、実施の形態１にかかる複合機１００には、複数のアプリケーション１３０および複数のコントロールサービスが存在し、いずれもプロセスとして動作している。そして、これらの各プロセス内部には、一又は複数のスレッドが

生成されて、スレッド単位の並列実行が行われる。そして、コントロールサービスがアプリケーション１３０に対し共通サービスを提供しており、このため、これらの多数のプロセスが並列動作、およびスレッドの並列動作を行って互いにプロセス間通信を行って協調動作をしながら、コピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるユーザサービスを提供するようになっている。また、複合機１００には、サードベンダなどの第三者がコントロールサービス層の上のアプリケーション層に新規アプリ１１７、１１８を開発して搭載することが可能となっている。図１では、この新規アプリ１１７、１１８を搭載した例を示している。

【００５０】

なお、実施の形態１にかかる複合機１００では、複数のアプリケーション１３０のプロセスと複数のコントロールサービスのプロセスとが動作しているが、アプリケーション１３０とコントロールサービスのプロセスをそれぞれ単一の構成とすることも可能である。また、各アプリケーション１３０は、アプリケーションごとに追加又は削除することができる。更に、実施の形態１にかかる複合機１００では、ＶＡＳ１４０に加えてダイナミックリンクライブラリ（ＤＬＬ）を採用することが可能である。

【００５１】

図２に複合機１００のハードウェア構成例を示す。

【００５２】

複合機１００は、コントローラ１６０と、オペレーションパネル１７５と、ファックスコントロールユニット（ＦＣＵ）１７６と、プリンタ等の画像形成処理に特有のハードウェア資源であるエンジン部１７７とを含む。コントローラ１６０は、ＣＰＵ１６１と、システムメモリ１６２と、ノースブリッジ（ＮＢ）１６３と、サウスブリッジ（ＳＢ）１６４と、ＡＳＩＣ１６６と、ローカルメモリ１６７と、ＨＤＤ１６８と、ネットワークインターフェースカード（ＮＩＣ）１６９と、ＳＤカード用スロット１７０と、ＵＳＢデバイス１７１と、ＩＥＥＥ１３９４デバイス１７２と、セントロニクス１７３とを含む。なお、メモリ１６２、１６７はＲＡＭ、ＲＯＭ等を含む。ＦＣＵ１７６およびエンジン部１７７は、コ

ントローラ１６０のＡＳＩＣ１６６にＰＣＩバス１７８で接続されている。

【００５３】

ＣＰＵ１６１が、複合機１００にインストールされるアプリケーション、コントロールサービス等のプログラムを、メモリから読み出して実行する。

【００５４】

図３は、実施の形態１にかかる複合機１００のＶＡＳ１４０の構成と、ＶＡＳ１４０と各アプリ、コントロールサービス層１５０および汎用ＯＳ１２１との関係を示すブロック図である。なお、図３では、アプリケーション１３０の例として、プリンタアプリ１１１、コピーアプリ１１２、新規アプリ１１７、１１８を示しているが、更にアプリを追加することもできる。

【００５５】

仮想アプリケーションサービス（ＶＡＳ）１４０のプロセスには、ディスパッチャ１４４と、制御スレッド１４３と、ラッピング処理スレッド１４１と、バージョン管理スレッド１４２とが動作している。

【００５６】

ディスパッチャ１４４は、アプリケーション１３０やコントロールサービスからのメッセージ受信を監視し、受信したメッセージに応じて制御スレッド１４３、ラッピング処理スレッド１４１、バージョン管理スレッド１４２に処理要求を行うものである。実施の形態１の複合機１００では、ディスパッチャ１４４は、各アプリが起動したときにアプリからのアプリ登録要求メッセージを受信すると、そのアプリ登録要求メッセージを制御スレッド１４３に送信する。

【００５７】

制御スレッド１４３は、ディスパッチャ１４４からアプリ登録要求メッセージを受信してアプリ登録処理を行う。ここで、アプリ登録処理とは、ＲＡＭ２１０にアプリ登録テーブル（図示せず）を生成し、アプリ登録要求メッセージを送信したアプリの識別情報であるアプリＩＤをアプリ登録テーブルに記録する処理をいう。また、制御スレッド１４３は、ＨＤ２００に格納されたラッピング処理情報ファイル２０１を参照して、アプリ登録要求を行ったアプリについて、ラッピング処理情報が記録されているか否かをチェックする。

【００５８】

ラッピング処理スレッド１４１の機能概要は次のとおりである。

【００５９】

コントロールサービスが提供する関数（ＡＰＩ）がバージョンアップして引数の追加等がされたが、アプリ側でそのコントロールサービスに対して使用するその関数はバージョンアップ前のものであったとしても、ラッピング処理スレッド１４１がアプリ側から呼び出された関数の変更を行うことにより、バージョン差を吸収するというものである。

【００６０】

更に、ラッピング処理スレッド１４１は、ラッピング処理情報ファイル２０１を参照することにより、コントロールサービス層１５０からアプリに対して通知されるメッセージを選択する機能を有している。特に、複合機１００のシステムに大きく影響を与えるインタフェースについては、サードベンダなどの第三者に対し隠蔽しておき、直接コントロールサービスに対してアクセスすることを回避する。これにより、複合機のセキュリティ面および障害発生の未然防止を行うことができる。

【００６１】

また、バージョン管理スレッド１４２は、制御スレッド１４３からの処理要求を受け、アプリがＶＡＳ１４０に対して使用する関数のバージョンと、実際のＶＡＳ１４０が有する関数のバージョンとの違いを検出し、バージョン管理テーブル２１１を参照することにより、発生したバージョン差がＶＡＳ１４０によってサポート可能な範囲か否かを判定する。

【００６２】

図４は、ＨＤ２００に格納されるラッピング処理情報ファイル２０１の内容例を示す説明図である。図４に示すように、ラッピング処理情報ファイル２０１は、コントロールサービス層１５０からアプリに対するメッセージ（イベント）の非通知設定を行うための情報ファイルであり、複合機１００のシステムに大きく影響を与えるインタフェースについては第三者に対して隠蔽するよう、メッセージごとに任意の非通知設定を行うことができる。図４のイベントＡとＣは、非通

知設定がなされているのでアプリには通知されず、また、イベントＢのように非通知設定がなされていないイベントについては、通常通り通知が行われる。

【００６３】

図５は、バージョン管理テーブル２１１の内容例を示す説明図である。なお、このテーブルは、ＶＡＳ１４０の実行ファイルに含める形で実装でき、ＶＡＳ１４０が実行される際には例えばＲＡＭ２１０に展開される。実行ファイルにテーブル２１１を含めるには、例えば、図５に示すバージョン情報をインクルードファイルとして作成しておき、作成したプログラムにインクルードしてコンパイルする。また、このテーブルを別ファイルとし、これを複合機１００に格納しておき、ＶＡＳ１４０が適宜このファイルを参照してもよい。

【００６４】

図５に示すように、このバージョン管理テーブル２１１には、ＶＡＳ１４０の全体バージョン番号と、ＶＡＳ１４０がサポート可能なアプリの全体バージョンの範囲が記録されている。なお、サポート可能な範囲は、例えば、複合機１００で使用されるＶＡＳ１４０のバージョンより一定範囲前のバージョンまでである。

【００６５】

なお、ここで、ＶＡＳ１４０の全体バージョンとは、ＶＡＳ１４０が提供する関数のセットのバージョンであり、アプリの全体バージョンとは、アプリがＶＡＳ１４０に対して使用する関数のセットのバージョンである。なお、アプリがＶＡＳ１４０に対して使用する関数のセットのバージョン、アプリを作成するときに使用したＶＡＳ１４０の関数セットのバージョンと同じである。

【００６６】

更に、バージョン管理テーブル２１１は、関数毎に、ＶＡＳ１４０の関数のバージョン番号とその関数がサポートできる、アプリ側がＶＡＳ１４０に対して使用する関数のバージョンの範囲を含んでいる。なお、ＶＡＳ１４０は上記のように各機能をスレッドとして実装してもよいし、１つのプロセスとして動作するようにしてもよい。

【００６７】

次に、このように構成された複合機１００のＶＡＳ１４０によるラッピング処理とバージョン管理について説明する。まず、ラッピング処理について説明する。

【００６８】

図６は、ＶＡＳ１４０のラッピング処理スレッドによるラッピング処理の手順を示すフローチャートである。

【００６９】

コントロールサービスが提供する関数がバージョンアップされた場合、その関数と、アプリがそのコントロールサービスに対して使用する関数との間にバージョン差が生じる。このような場合に、上記関数間の違いを吸収するためのＶＡＳ１４０の処理を図６を用いて説明する。

【００７０】

なお、ＶＡＳ１４０は、各コントロールサービスに対する関数毎に、その前のバージョンに対して、関数がバージョンアップされているか否か、バージョンアップされているとすればそのバージョンアップがどのようなものかを示す情報を有している。この情報は、ＶＡＳ１４０のプログラムの作成時に、例えば図７に示すテーブルの形式でプログラムに含めることができる。なお、バージョンアップの態様としては、例えば、関数の追加、分割、削除、変更、統合があり、また、引数の追加、分割、削除がある。

【００７１】

まず、ステップＳ６０１において、アプリからの処理要求に対応する関数がバージョンアップされているか否かを図７の情報を参照することによりチェックする。バージョンアップされていなければ通常の処理を行う（ステップＳ６０４）。バージョンアップされていれば、そのバージョンアップの態様が、関数の分割又は引数の追加かどうかを判断する（ステップＳ６０２）。

【００７２】

ここで、関数の分割又は引数の追加、のいずれでもない場合には、バージョンアップが関数の削除、変更、統合等である。この場合には、アプリからの要求に対応する関数がコントロールサービス側に存在しなくなるため、アプリからの関

数呼び出しがあってもその処理を行うことができない。このため、アプリに対してＮＧであることを通知し（ステップＳ６０５）、終了を処理する（ステップＳ６０６）。

【００７３】

一方、ステップＳ６０２において、関数の分割、引数の追加である場合、対応する関数の数、引数の数が増えるため、増えた部分にダミー関数、又は、ダミー引数を補填する（ステップＳ６０３）、そして、通常の処理を行う（ステップ６０４）。なお、ダミー引数を補充する場合のＶＡＳ１４０の記述例を図８に示す。図８に示すように、アプリ側から要求を受ける関数“API_for_Apli_Nol(int arg1, int arg2)”を、引数dummyを含む“API_for_XCS_Nol(arg1, arg2, dummy)”に変換している。

【００７４】

なお、関数が追加されたことによりコントロールサービスがバージョンアップしている場合には、既存の関数自体は変化しないことから、追加された機能はアプリから使えないが、アプリからの関数呼び出しには影響を与えない。

【００７５】

次に、コントロールサービスの特定のインタフェースを隠蔽するための処理について説明する。コントロールサービスからアプリに対して全てメッセージ通知（例えば、イベント通知）が行われるとすると、複合機１００のシステムに大きく影響を与えるコントロールサービスに対して直接アクセスすることが可能となり、複合機のセキュリティ面および障害発生を未然に防止する面から好ましくない。

【００７６】

そこで、実施の形態１では、非通知とするイベントを、図４のラッピング処理情報ファイル２０１に予め設定する。この設定はどのようにして行ってもよい。例えば、複合機１００外でラッピング処理情報ファイル２０１を作成し、複合機１００に格納してもよいし、ＶＡＳ１４０が設定のための画面をオペレーションパネルに表示し、そこから設定してもよい。また、ＶＡＳ１４０の実行ファイルに含めるようにして実装してもよい。

【００７７】

図９は、ＶＡＳ１４０によりラッピング処理を行う手順を示すフローチャートである。

【００７８】

あるコントロールサービスからアプリに向けてメッセージ通知がなされる場合、ラッピング処理スレッド１４１がラッピング処理情報ファイル２０１を参照する（ステップＳ７０１）。そのメッセージに対して非通知設定がなされているかどうかをチェックし（ステップＳ７０２）、そのメッセージに対して非通知設定がなされていれば、ラッピング処理スレッド１４１はそのメッセージをアプリに通知しない（ステップＳ７０３）。また、上記ステップＳ７０１において、上記のメッセージに対してラッピング処理情報ファイル２０１内の設定自体が行われていないか、非通知設定されていない場合は、通常通りアプリにメッセージが通知される（ステップＳ７０４）。

【００７９】

ここでは、特定のイベントをアプリに通知するか否かを任意かつ事前に設定できるようにしたため、複合機のセキュリティ面および障害発生を未然に防止することが可能になる。また、実施の形態１では、新規アプリケーションを開発するサードベンダなどの第三者に特定のインタフェースを開示する必要があるため、開示できるインタフェースを自由に選択できるようにすることは非常に重要である。

【００８０】

次に、アプリがＶＡＳ１４０に対して使用する関数（ＡＰＩ）と、ＶＡＳ１４０が有する関数とのバージョン差をチェックする手順について説明する。上記のように、ＶＡＳ１４０により、アプリとコントロールサービス間での関数のバージョン差を吸収するが、ＶＡＳ１４０自体のバージョンアップにより、ＶＡＳ１４０がアプリに対して提供する関数と、アプリがＶＡＳ１４０に対して使用する関数との間でバージョン差が生じることがある。これから説明する処理は、バージョン差をチェックし、ＶＡＳ１４０がサポートできる範囲のバージョン差であればＶＡＳ１４０の通常の処理を行うというものである。

【００８１】

この処理では、アプリ自身が、全体バージョン情報と、ＶＡＳ１４０に対して使用する関数毎のバージョン情報を保持し、アプリが起動したときに、その情報をプロセス間通信によりＶＡＳ１４０に通知している。アプリ側のこのバージョン情報は、例えばアプリの作成時にアプリに含められる。

【００８２】

ＶＡＳ１４０によるバージョンチェック時期としては、アプリが実行される前であれば良く、例えばここではアプリ登録時に行ったが、これに限定されない。例えば、アプリを仮起動させて、そのときにＶＡＳ１４０がバージョンチェックを行ってもよい。仮起動とは、ＶＡＳ１４０が、アプリとのプロセス間通信によりアプリの情報を取得するためだけにアプリを起動することである。なお、この場合のアプリは仮起動可能なように作られている。

【００８３】

登録時には次のような処理が行われる。

【００８４】

ディスパッチャ１４４が、起動されたアプリからアプリ登録要求メッセージを受信すると、アプリ登録要求メッセージをそのアプリのプロセスＩＤとともに制御スレッド１４３に受け渡す。制御スレッド１４３は、アプリ登録要求メッセージとプロセスＩＤをディスパッチャ１４４から受信すると、アプリを識別するアプリＩＤを決定して、ＲＡＭ２１０などに格納したアプリ登録テーブル（図示せず）にアプリＩＤを記録することにより、アプリ登録が行われる。なお、アプリＩＤは、コピーアプリ１１２、プリンタアプリ１１１など既存のアプリケーションについては、予め定められており、各アプリＩＤをＶＡＳ１４０の内部で保持している。また、サードベンダなどが開発した新規アプリ１１７、１１８については、最初の起動時におけるアプリ登録処理の中で決定される。

【００８５】

以下、全体バージョンのチェックを行う場合と、関数毎のバージョンチェックを行う場合を分けて説明するが、全体バージョンチェックにおいてサポート範囲外と判断された場合にのみ、関数毎のバージョンチェックを行うようにしてもよ

い。

【００８６】

まず、全体バージョンのチェックを行う場合について図１０を参照して説明する。

【００８７】

アプリ登録の要求があった場合（ステップＳ８０１）、まず、ＶＡＳ１４０はアプリが有している全体バージョン番号を取得する（ステップＳ８０２）。

【００８８】

ＶＡＳ１４０は、バージョン管理テーブル２１１の全体バージョンのサポート範囲とアプリの全体バージョン番号とを比較し、サポート範囲内か否かを判断する（ステップＳ８０３）。例えば、現在使用されているＶＡＳ１４０がサポートできるバージョンの範囲が１．０～１．６であれば、この範囲内のバージョン番号であるときにサポート範囲内と判定する。この場合、当該アプリに対して「ＯＫ」が通知され（ステップＳ８０４）、正常にアプリ登録され（ステップＳ８０５）、通常の複合機１００における処理が行われる（ステップＳ８０６）。また、ステップＳ８０３でアプリから取得したバージョン番号がサポート範囲外である場合には、当該アプリに対して「ＮＧ」が通知され（ステップＳ８０７）、アプリ登録が終了する（ステップＳ８０８）。

【００８９】

次に、バージョンを関数毎にチェックする場合について説明する。図１１にその処理手順を示す。

【００９０】

図１１の場合も、図１０と同様にＶＡＳ１４０によるバージョンチェック時期として、アプリが実行される前であれば良く、ここではアプリ登録時を想定している。

【００９１】

アプリ登録の要求があった場合（ステップＳ９０１）、ＶＡＳ１４０はアプリから、アプリがＶＡＳ１４０に対して使用している関数毎のバージョン情報を取得する（ステップＳ９０２）。そして、ＶＡＳ１４０は、バージョン管理テーブ

ル２１１を参照し、アプリがＶＡＳ１４０に対して使用している関数毎に、アプリがＶＡＳ１４０に対して使用する関数のバージョン番号と、サポート範囲とを比較することにより、当該関数がＶＡＳ１４０のサポート範囲内か否かを判断する（ステップＳ９０３）。例えば、アプリがＶＡＳ１４０に対して使用している関数番号が「２」、「３」であり、そのバージョン番号がそれぞれ「バージョン１．１」と「バージョン２．０」である場合には、各関数がサポート範囲内となる。

【００９２】

図１１の処理において、ステップＳ９０３においてアプリケーションがＶＡＳ１４０に対して使用している全ての関数についてサポート範囲内と判定された場合は、上記した図１０のＡ以降と同様に処理され、ステップＳ９０３においてサポート範囲内でないと判定された場合は、図１０のＢ以降と同様に処理される。なお、ＮＧの場合に、ＶＡＳ１４０は、どの関数がＮＧであるかをオペレーションパネルに表示してもよい。

【００９３】

アプリが使用していない関数のバージョンがＶＡＳ１４０側で変更された場合でも、ＶＡＳ1.４０のバージョンは変更されるため、全体バージョンの比較をしただけではバージョン差によってアプリの実行が不可となる場合があり得る。そこで、図１１に示した処理のように、アプリがＶＡＳ１４０に対して使用している関数のバージョンを関数毎にチェックすることにより、上記の場合でもアプリの実行に問題がないことを判断できる。また、関数毎のチェックでサポート範囲外と判定された場合、その原因となった関数を容易に特定できる。

【００９４】

このように、実施の形態１にかかる複合機１００では、ＶＡＳ１４０のラッピング処理スレッド１４１が、バージョン差を吸収するので、コントロールサービスの関数がバージョンアップしたとしても、アプリをコンパイルし直す必要が無くなる。また、ＶＡＳ１４０は、コントロールサービスからアプリに対するメッセージを取捨選択できるため、複合機１００のシステムに大きく影響を与えるようなインタフェースについては、第三者に対して隠蔽することができる。

【００９５】

また、実施の形態１にかかる複合機１００では、ＶＡＳ１４０が、アプリから取得したバージョン情報を用いてバージョンチェックをできるので、安全にアプリを実行できる。また、実施の形態１にかかる複合機１００では、上記したようにＶＡＳ１４０に加えてダイナミックリンクライブラリ（ＤＬＬ）を採用することも可能であり、その場合はアプリケーションとコントロールサービス間のバージョン差の吸収力が増大するとともに、インタフェースを隠蔽することができるためインタフェースの秘匿性をより強固に確保することが可能となる。

【００９６】

（実施の形態２）

実施の形態１にかかる複合機１００は、ＶＡＳ１４０が全アプリケーションに対して１つだけ存在するものであった。この実施の形態２にかかる複合機では、各アプリごとに一つのＶＡＳが起動し、各ＶＡＳは対応するアプリに対してのみバージョン管理およびラッピング処理を行うものである。

【００９７】

図１２は、実施の形態２にかかる複合機８００の構成を示すブロック図である。図１２に示すように、複合機８００では、複数の仮想アプリケーションサービス（ＶＡＳ）８４１～８４８がアプリケーション１３０の各アプリごとに動作している点が、実施の形態１にかかる複合機１００と異なっている。

【００９８】

ＶＡＳ８４１～８４８は、プリンタアプリ１１１、コピーアプリ１１２、ファックスアプリ１１３、スキャナアプリ１１４、ネットファイルアプリ１１５、工程検査アプリ１１６、新規アプリ１１７および１１８に対応して、バージョン管理およびラッピング処理を行うようになっている。

【００９９】

図１３は、実施の形態２にかかる複合機８００のＶＡＳ８４１～８４８の構成と、ＶＡＳ８４１～８４８と各アプリ、コントロールサービス層１５０および汎用ＯＳ１２１との関係を示すブロック図である。なお、図１２では、アプリケーション１３０として、プリンタアプリ１１１、コピーアプリ１１２、新規アプリ

１１７、１１８の例を示し、更にこれら各アプリに対応したＶＡＳ８４１、８４２、８４７および８４８を例として示したが、他のアプリの場合も同様の構成となる。

【０１００】

また、実施の形態２にかかる複合機８００では、実施の形態１の複合機１００と異なり、図１３に示すように、各ＶＡＳ８４１～８４８と各アプリとの間にはＶＡＳ制御プロセス（デーモン）８０１が動作している。

【０１０１】

このＶＡＳ制御プロセス（デーモン）８０１は、各アプリからアプリ登録要求メッセージを受信してアプリ登録処理を行うとともに、アプリ登録要求を行ったアプリに対応したＶＡＳ８４１～８４８を生成する。また、各スレッドに対して処理の指示を行う。

【０１０２】

そして、仮想アプリケーションサービス（ＶＡＳ）８４１～８４８のプロセスには、ディスパッチャ１４４と、ラッピング処理スレッド１４１と、バージョン管理スレッド１４２とが動作している。

【０１０３】

ディスパッチャ１４４は、アプリケーション１３０やコントロールサービスからのメッセージ受信を監視し、例えば、受信したメッセージに応じてラッピング処理スレッド１４１、あるいはバージョン管理スレッド１４２に対して処理要求を行うものである。実施の形態２の複合機８００では、ディスパッチャ１４４は、ＶＡＳ制御プロセス８０１から、アプリＩＤ、アプリのプロセスＩＤとともに、バージョン管理要求メッセージ又はラッピング処理要求メッセージを受信するようになっている。ディスパッチャ１４４は、バージョン管理要求メッセージを受信したときには、アプリＩＤ、アプリのプロセスＩＤとともに受信したバージョン管理要求メッセージをバージョン管理スレッド１４２に送信し、ラッピング処理要求メッセージを受信したときには、アプリＩＤ、アプリのプロセスＩＤとともに受信したラッピング処理要求メッセージをラッピング処理スレッド１４１に送信する。バージョン管理スレッド１４２とラッピング処理スレッド１４１の

処理内容は第１の実施の形態と同様である。

【０１０４】

このように実施の形態２にかかる複合機８００によれば、第１の実施の形態と同様の効果がある。更に、実施の形態２にかかる複合機８００では、ＶＡＳ８４１～８４８は起動されるアプリケーション１３０ごとに別個に起動されるので、複数のアプリケーション１３０に対する判断処理を各アプリケーション１３０に対応するＶＡＳ８４１～８４８で並列に実行することができ、処理が効率的に行えると共に、上記したバージョン管理やラッピング処理がアプリごとに行えるので、新規アプリの追加や交換時にも効率良く処理を行うことができる。

【０１０５】

また、ＶＡＳの構成として上記の構成の他、図１４（ａ）～（ｃ）に示す構成をとることもできる。

【０１０６】

図１４（ａ）は、各アプリケーションに対して起動されるＶＡＳを、親ＶＡＳの子プロセスとする場合であり、親ＶＡＳ自体は画面制御権（ユーザインターフェース）を持たない。図１４（ｂ）は、親ＶＡＳ自体が画面制御権（ユーザインターフェース）を持つ場合である。図１４（ｃ）は、各アプリケーションに対応するＶＡＳの機能をスレッドとして起動する場合を示している。

【０１０７】

なお、本発明は、上記の実施の形態に限定されることなく、特許請求の範囲内において、種々変更・応用が可能である。

【０１０８】

【発明の効果】

上記のように、本発明によれば、アプリケーションが使用している関数とコントロールサービス側の関数とで差異が生じた場合でも、アプリケーションのソースプログラムを書き換えることなく、関数呼び出しを行うことが可能となる。また、コントロールサービスからアプリに通知されるメッセージのうち特定のメッセージを秘匿することが可能となる。

【図面の簡単な説明】

【図１】

実施の形態１にかかる複合機の構成を示すブロック図である。

【図２】

実施の形態１にかかる複合機のハードウェア構成を示すブロック図である。

【図３】

実施の形態１にかかる複合機のＶＡＳの構成と、ＶＡＳと各アプリ、コントロールサービス層および汎用ＯＳとの関係を示すブロック図である。

【図４】

ＨＤに格納されるラッピング処理情報ファイルの内容例を示す説明図である。

【図５】

バージョン管理テーブルの内容例を示す説明図である。

【図６】

ＶＡＳによりラッピング処理を行う手順を示すフローチャートである。

【図７】

バージョンアップの有無及びバージョンアップの態様を記録したテーブルを示す図である。

【図８】

ダミー引数を補充する場合のＶＡＳ１４０の記述例を示す図である。

【図９】

ＶＡＳにより特定のメッセージを隠蔽するためのラッピング処理を行う手順を示すフローチャートである。

【図１０】

ＶＡＳにより全体バージョンをチェックする手順を示すフローチャートである。

【図１１】

ＶＡＳにより関数単位にバージョンをチェックする手順を示すフローチャートである。

【図１２】

実施の形態２にかかる複合機の構成を示すブロック図である。

【図１３】

実施の形態２にかかる複合機のＶＡＳの構成と、ＶＡＳと各アプリ、コントロールサービス層および汎用ＯＳとの関係を示すブロック図である。

【図１４】

ＶＡＳの構成例を示す図である。

【符号の説明】

１００　複合機

１０１　白黒レーザプリンタ

１０２　カラーレーザプリンタ

１０３　ハードウェアリソース

１１０　ソフトウェア群

１１１　プリンタアプリ

１１２　コピーアプリ

１１３　ファックスアプリ

１１４　スキャナアプリ

１１５　ネットファイルアプリ

１１６　工程検査アプリ

１１７、１１８　新規アプリ

１２０　プラットホーム

１２１　汎用ＯＳ

１２２　ＳＣＳ

１２３　ＳＲＭ

１２４　ＥＣＳ

１２５　ＭＣＳ

１２６　ＯＣＳ

１２７　ＦＣＳ

１２８　ＮＣＳ

１３０　アプリケーション

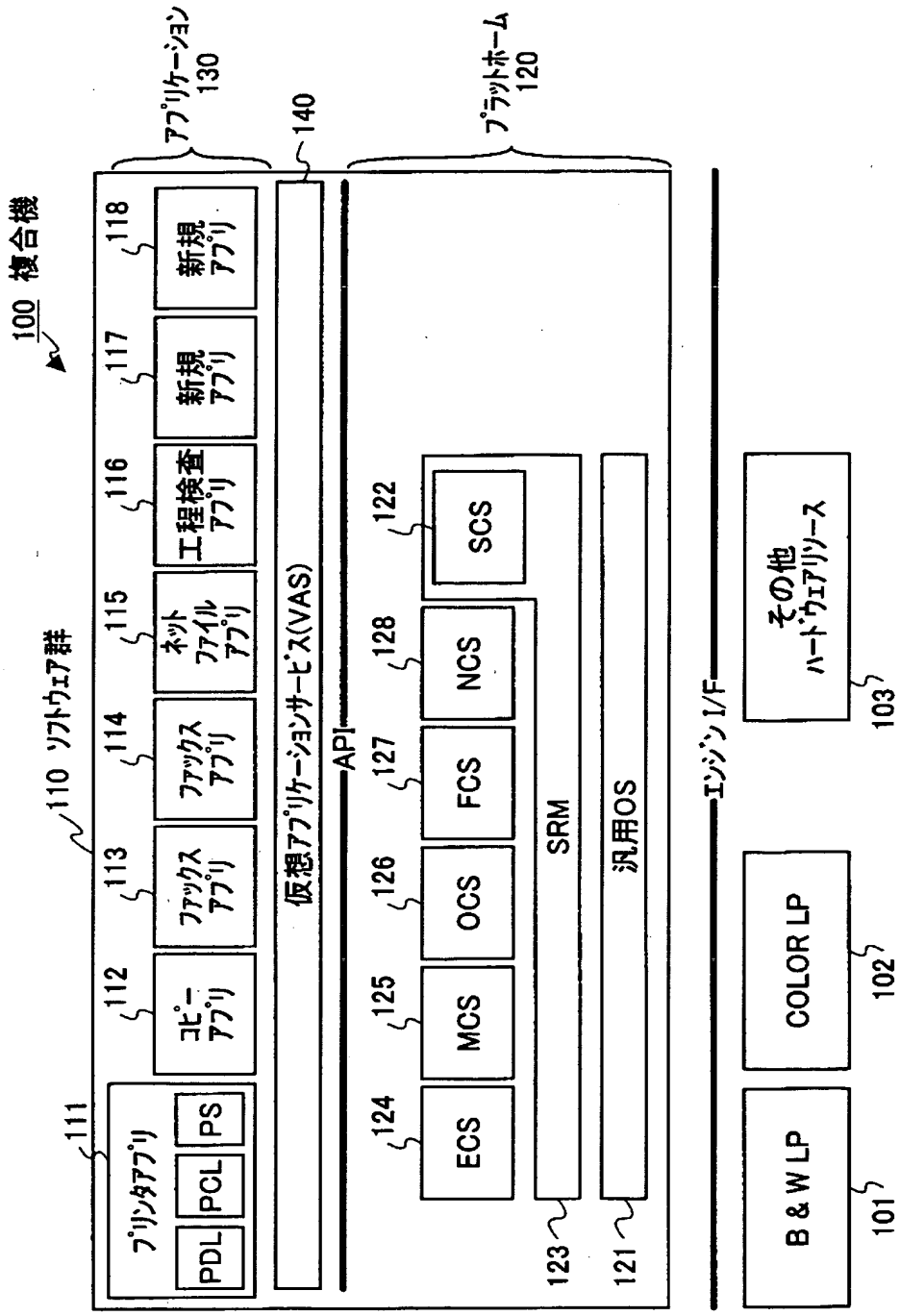
１４０、８４１～８４８　仮想アプリケーションサービス（ＶＡＳ）

１４１　ラッピング処理スレッド
１４２　バージョン管理スレッド
１４３　制御スレッド
１４４　ディスパッチャ
１５０　コントロールサービス層
２００　ハードディスク（ＨＤ）
２０１　ラッピング処理情報ファイル
２１０　ＲＡＭ
２１１　バージョン管理テーブル
８００　複合機
８０１　ＶＡＳ制御プロセス

【書類名】　図面

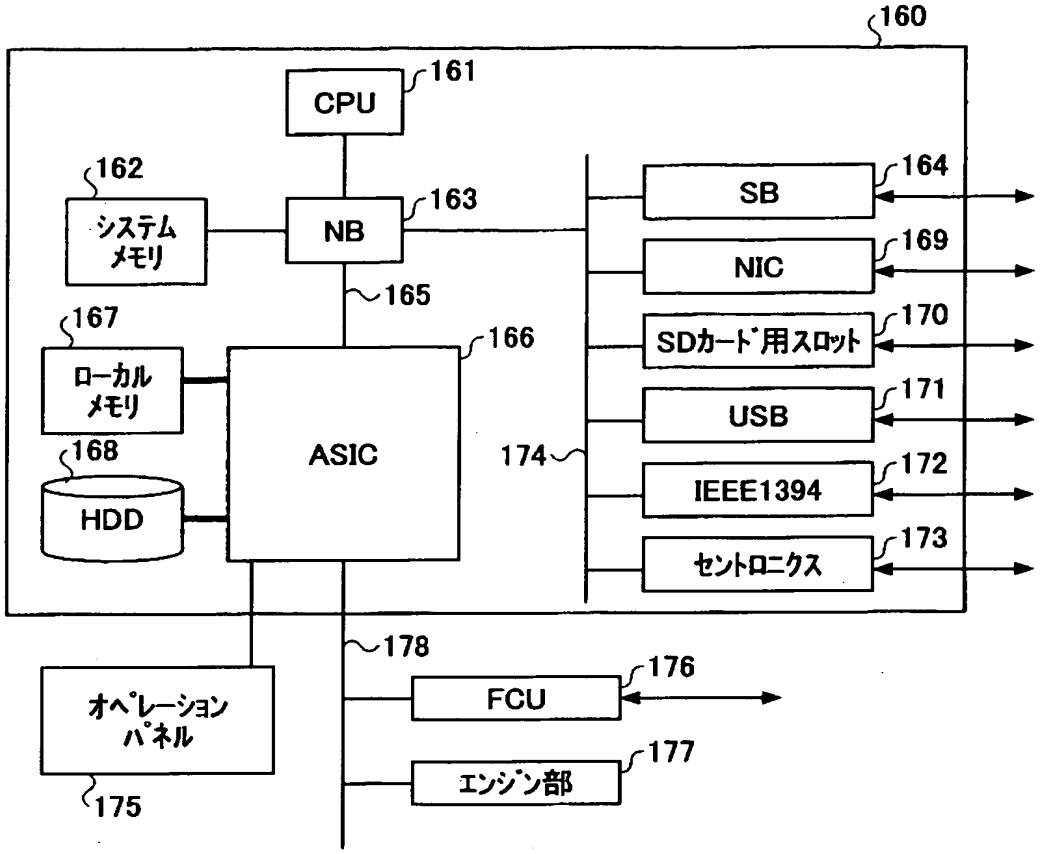
【図１】

実施の形態1にかかる複合機の構成を示すブロック図



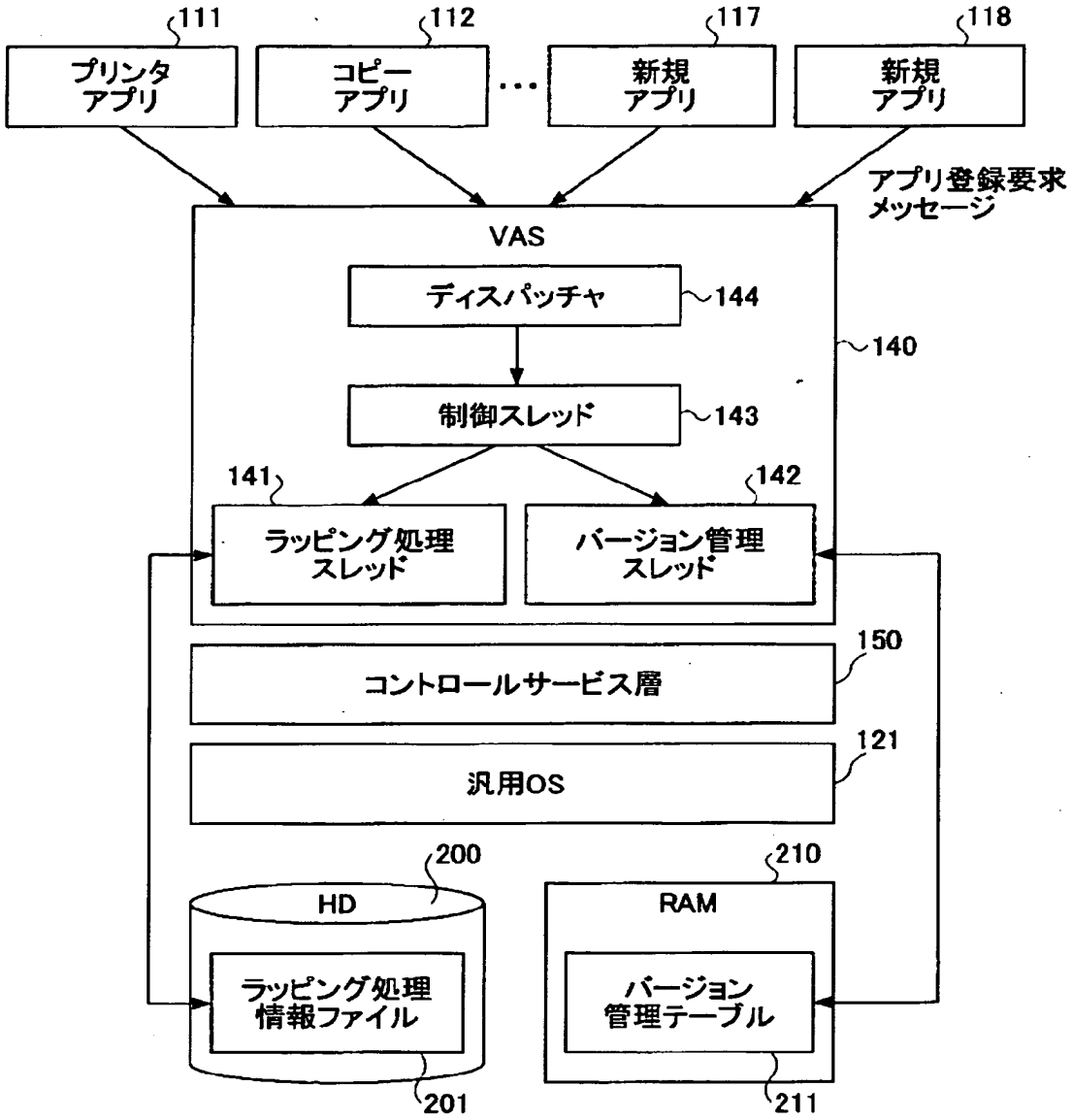
【図２】

実施の形態1にかかる複合機の
ハードウェア構成を示すブロック図



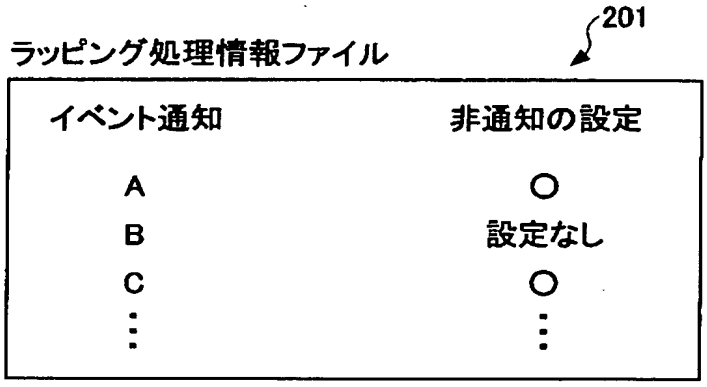
【図３】

実施の形態1にかかる複合機のVASの構成と、VASと各アプリ、コントロールサービス層および汎用OSとの関係を示すブロック図



【図４】

HDに格納されるラッピング処理情報ファイルの内容例を示す説明図



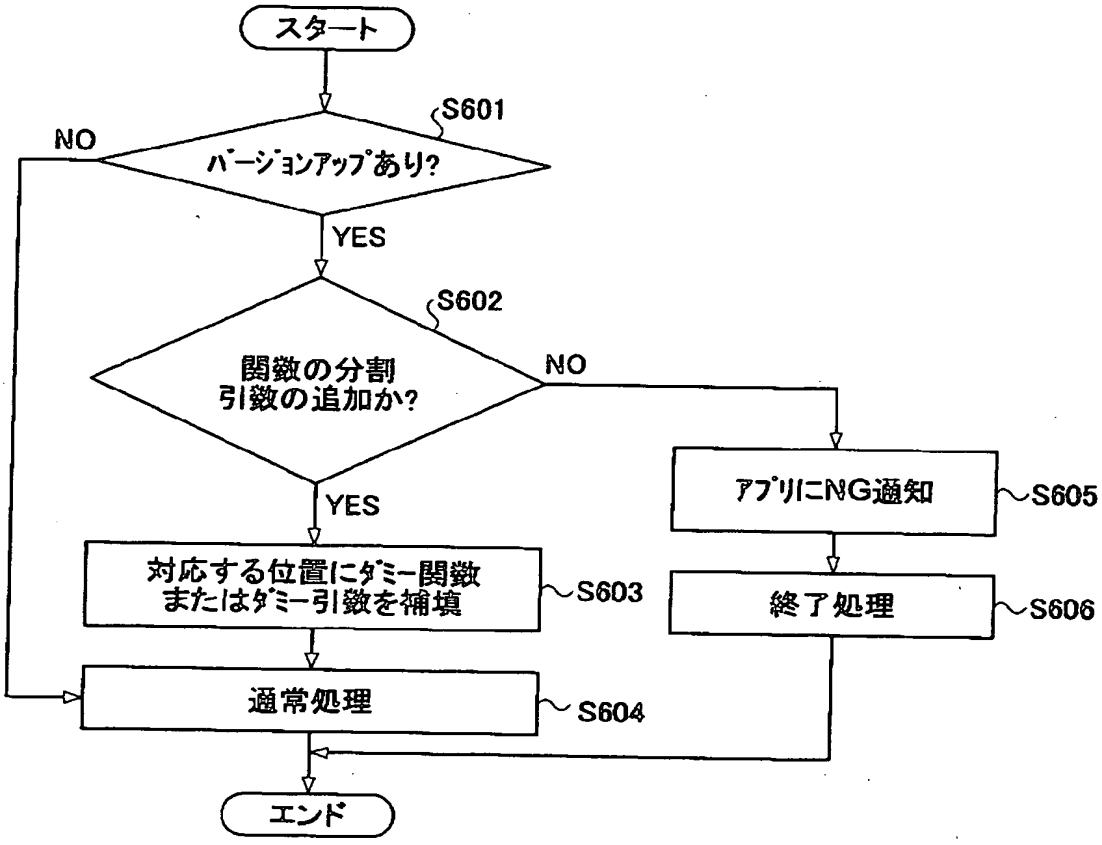
【図５】

バージョン管理テーブルの内容例を示す説明図

全体バージョン番号2.3		サポート範囲2.0～2.3	
関数番号	バージョン番号	サポート範囲	使用の有無
1	1.4	1.0～1.4	×
2	1.2	1.0～1.2	○
3	2.3	2.0～2.3	○
4	1.1	1.0～1.1	×
⋮	⋮	⋮	⋮
n			

【図６】

VASによりラッピング処理を行う手順を示すフローチャート



【図７】

バージョンアップの有無及び
バージョンアップの態様を記録したテーブルを示す図

関数番号	ﾊﾞｰｼﾞｮﾝ変更有無	関数分割	引数追加	・・・
1	1	0	1	
2	0	0	0	
3	0	0	0	
4	1	1	0	
⋮	⋮	⋮	⋮	

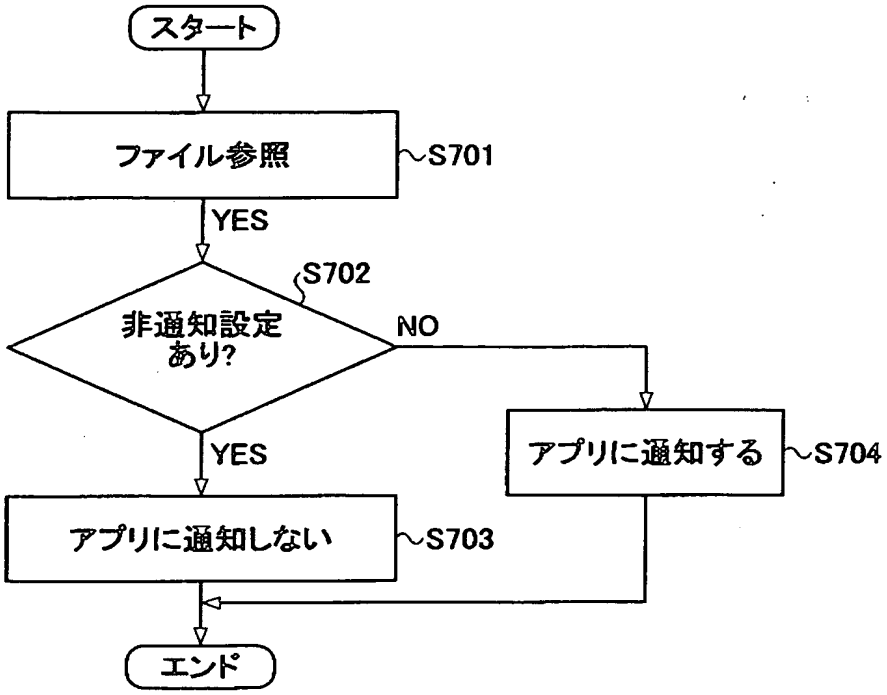
【図８】

ダミー引数を補充する場合のVAS140の記述例を示す図

```
API_for_Apli_No1(int arg1, int arg2) {
    int dummy = 0;
    ...
    API_for_XCS_No1(arg1, arg2, dummy);
    ...
}
```

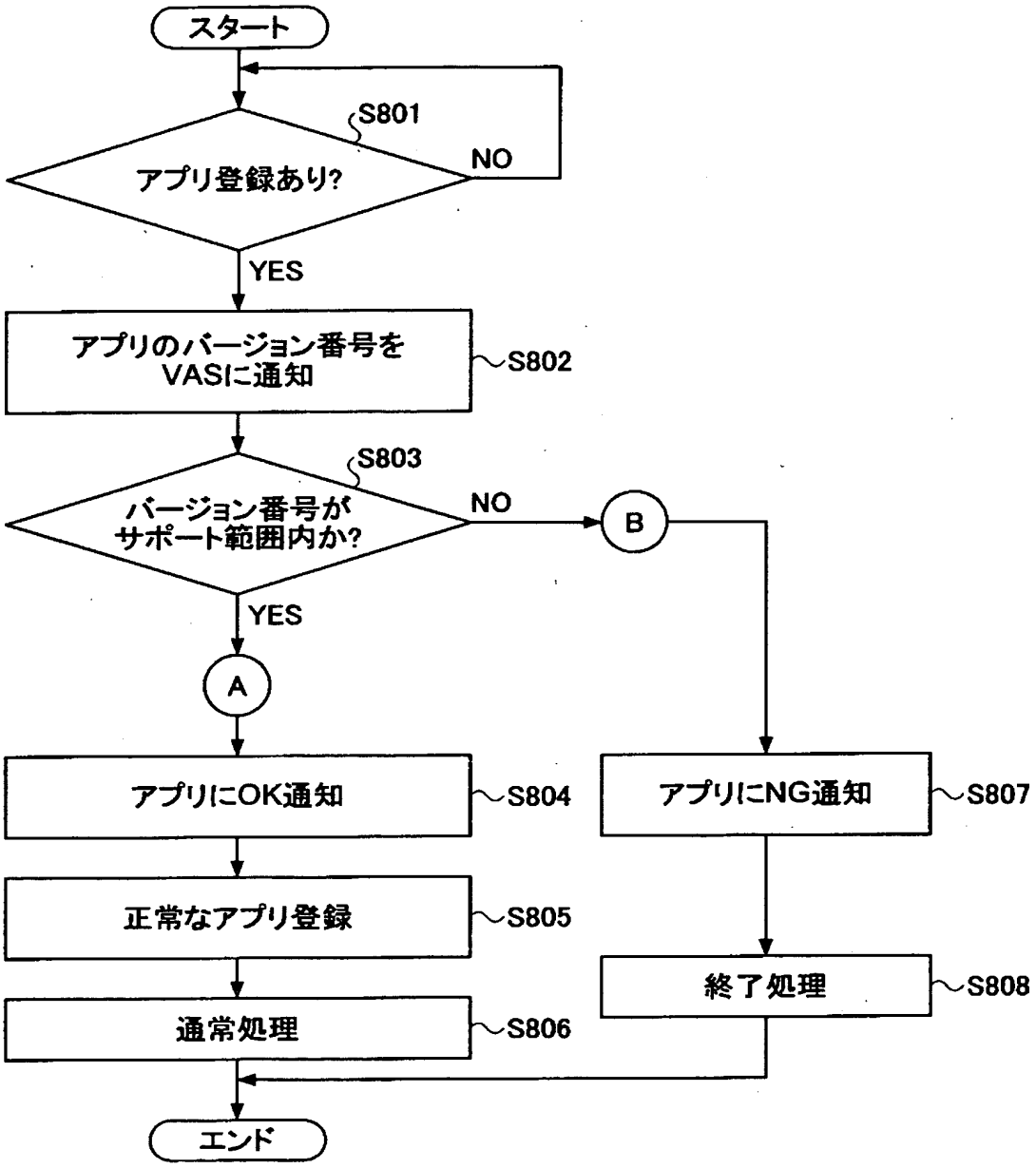
【図９】

VASにより特定のメッセージを隠蔽するための
ラッピング処理を行う手順を示すフローチャート



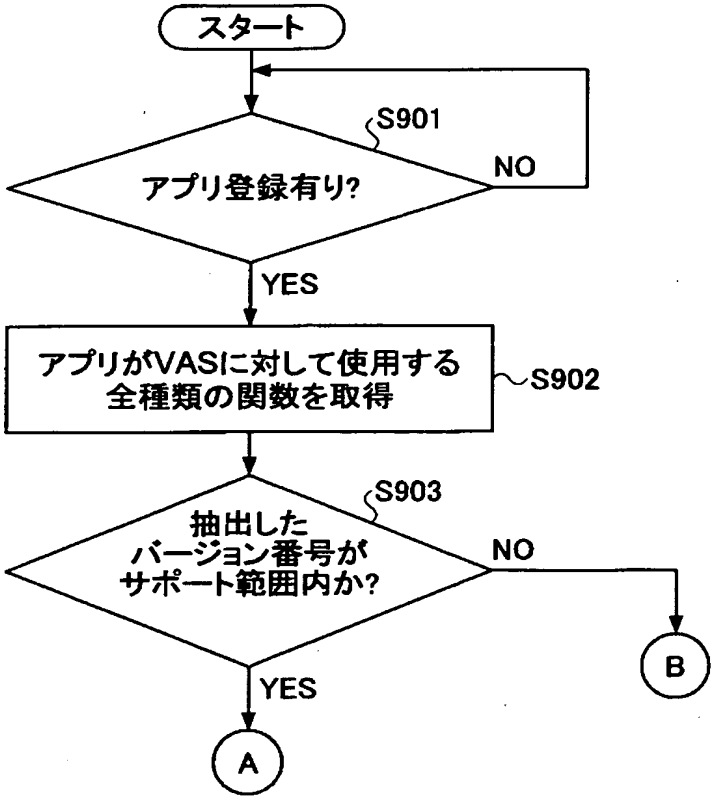
【図１０】

VASにより全体バージョンをチェックする手順を示すフローチャート



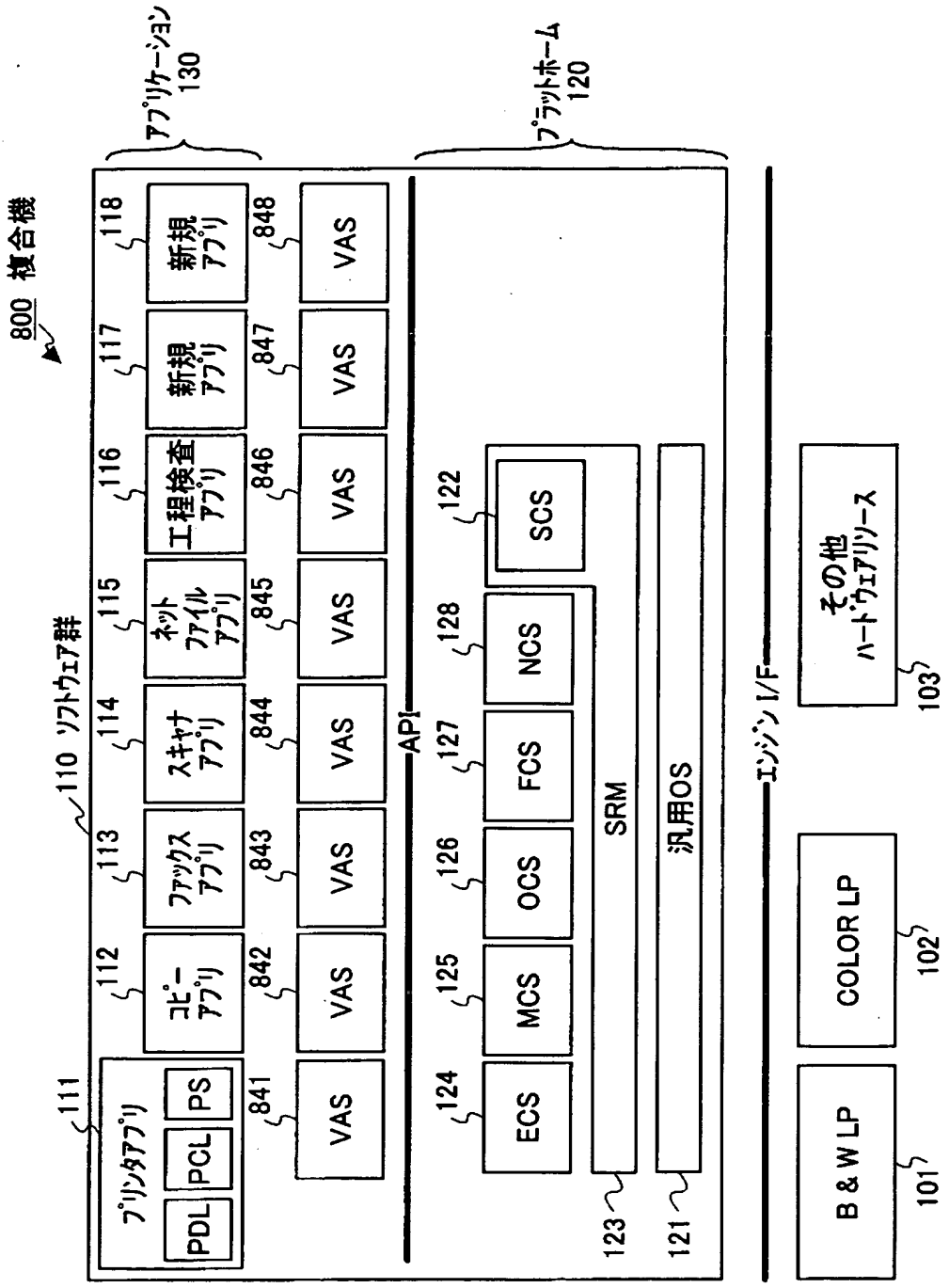
【図１１】

VASにより関数単位にバージョンをチェックする手順を示すフローチャート



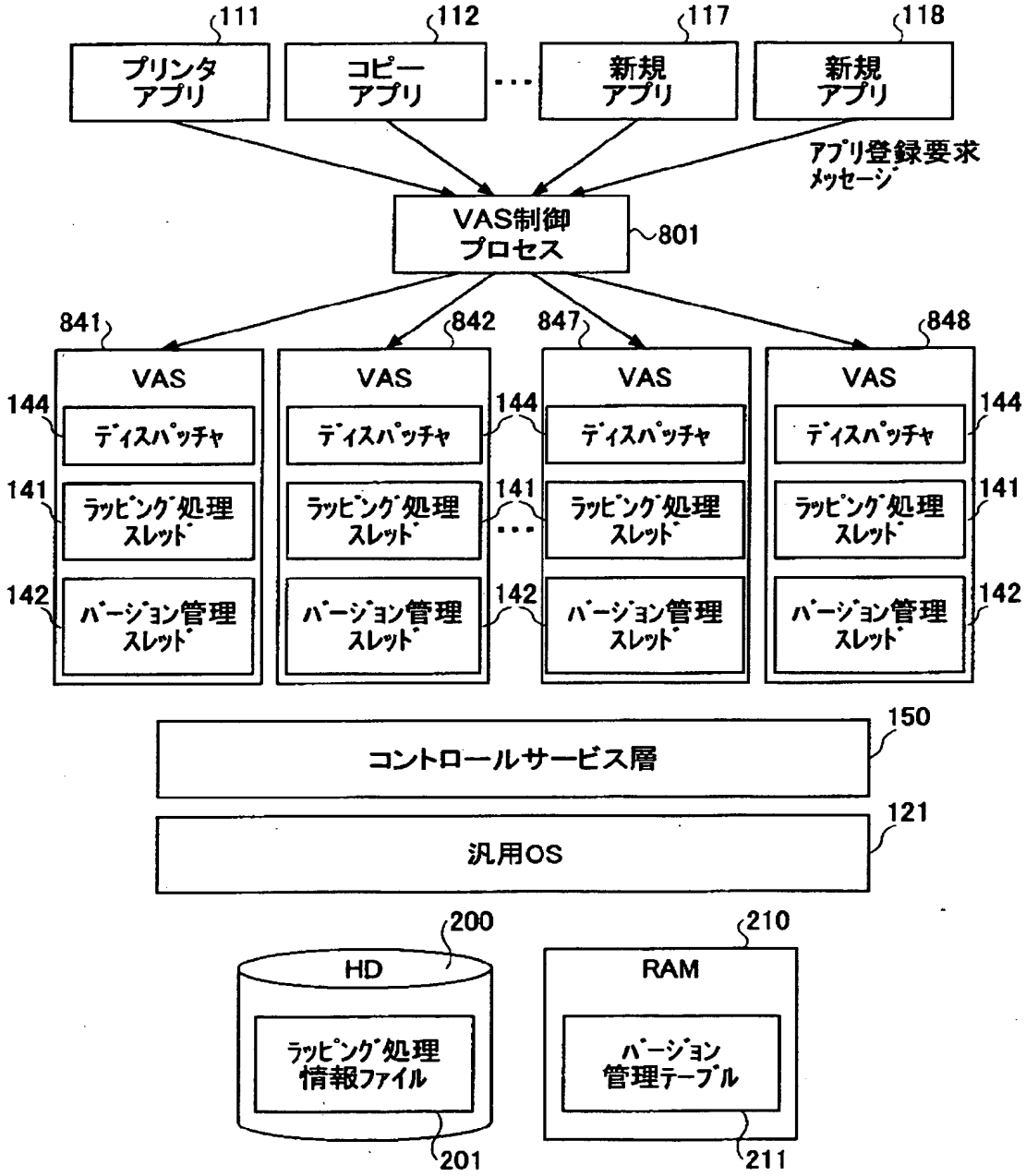
【図１２】

実施の形態2にかかる複合機の構成を示すブロック図



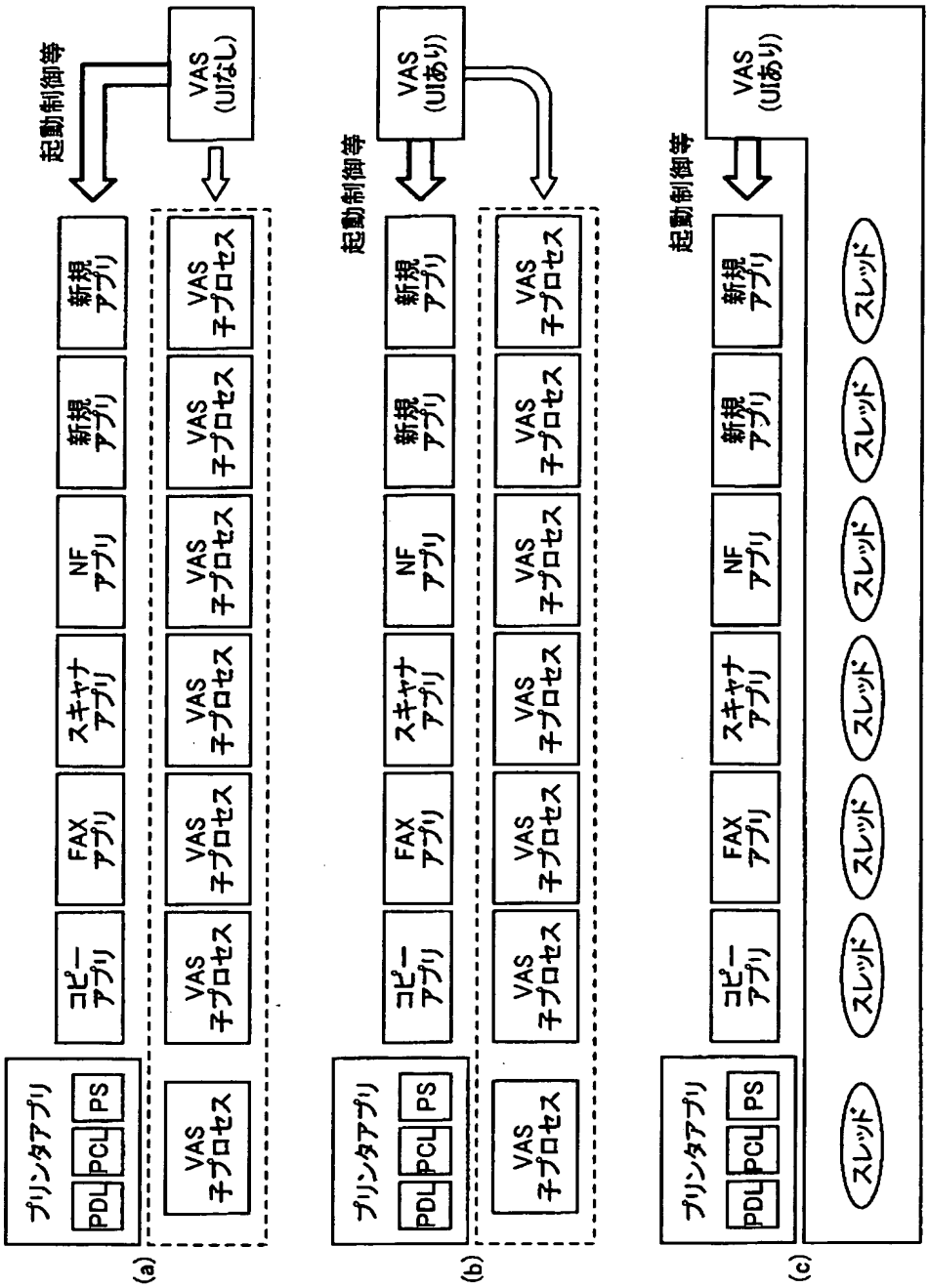
【図１３】

実施の形態2にかかる複合機のVASの構成と、VASと各アプリ、コントロールサービス層および汎用OSとの関係を示すブロック図



【図１４】

VASの構成例を示す図



【書類名】　　要約書

【要約】

【課題】　アプリケーションが使用している関数とコントロールサービス側の関数とで差異が生じた場合でも、関数呼び出しを行うことを可能とする技術を提供する。

【解決手段】　画像形成に関する処理を行うアプリケーションと、当該アプリケーションからの関数呼び出しに基づきシステム側の処理を行うコントロールサービスとを備えた画像形成装置において、アプリケーションから呼び出される関数を変換し、変換後の関数を用いてコントロールサービスへの関数呼び出しを行うラッピング処理手段を備える。

【選択図】　図３

特願２００３－１９６２２８

出　願　人　履　歴　情　報

識別番号　　［０００００６７４７］

１．変更年月日　　１９９０年　８月２４日
［変更理由］　　新規登録
住　所　　東京都大田区中馬込１丁目３番６号
氏　名　　株式会社リコー

２．変更年月日　　２００２年　５月１７日
［変更理由］　　住所変更
住　所　　東京都大田区中馬込１丁目３番６号
氏　名　　株式会社リコー